IN THE SPECIFICATION

Please amend paragraph 30 as follows:

[0030] **Figure 4** illustrates the general architecture of enhanced host controller interface (EHCI) 400. EHCI 400 comprises three interface spaces: peripheral component interconnect (PCI) configuration 410, register 420, and schedule interface 430. PCI configuration 410 includes PCI registers used for system component enumeration and PCI power management. PCI configuration registers in PCI configuration 410 comprise PCI class code 411, USB base address 412, and PCI power management interface 413. Register 420 comprises memory based input/output (I/O) registers. Memory based I/O registers are comprised of capability registers 421 and operational registers 422. Register 420 must be implemented as memory-mapped I/O. Schedule interface 430 is typically memory allocated and managed by the HC driver for the periodic list 431 and asynchronous schedules. EHCI 400 allows software to enable or disable each schedule.

Please amend paragraph 31 as follows:

[0031] There are typically two (2) methods for organizing qTDs. **Figure 5** illustrates the first method 500. In the method illustrated in **Figure 5**, all of the qTDs 510 necessary to represent at least two buffers are created. Each alternate-pointer of buffer N 520 points to the first qTD 510 of buffer N+1 530. This method, however, requires a large memory footprint to initialize all of the qTDs 510 required to represent both buffers.

Please amend paragraph 32 as follows:

[0032] **Figure 6** illustrates the second method 600. The second method initializes all the alternate-pointers of the qTDs 510 to a "dummy" qTD. When a short packet is received, the HC will vector to dummy qTD 620. Each alternate-pointer of buffer N 620 points to the first qTD 610 of buffer N+1 630. Software then detects the short packet and re-initializes qTDs 610. The software can only detect a short packet condition when the hardware asserts an interrupt. Since interrupts occur at fixed intervals, the time after the short packet is received and before the interrupt is serviced is unused. Therefore, this second approach, while having a small memory footprint, has low throughput.